Secure Multiparty Computation

Secret Sharing

Report prepared for Blockchain and Cryptocurrencies – AY 2022/2023

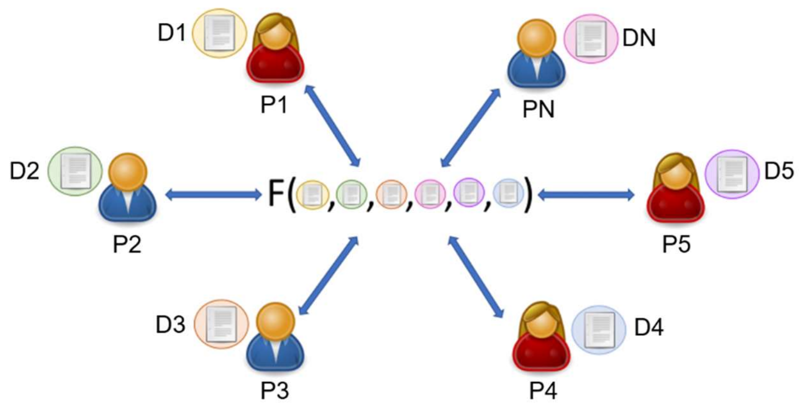
SUMMARY

1. Introduction

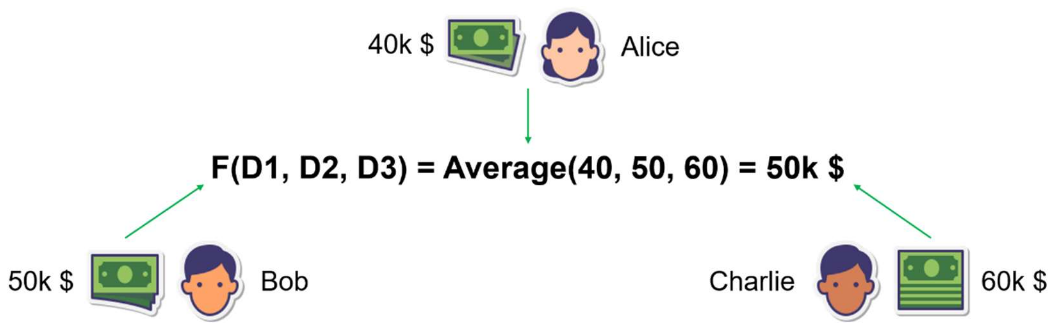
In the last few decades, data privacy and security has become the primary concern to everyone. Seven million data records are compromised every day. This problem has a global cost of $3.26 million. One of the techniques which provide the solution to the problems of data security and data privacy is Secure Multiparty Computation.

2. Secure Multiparty Computation

The Secure Multiparty Computation (SMPC) is a set of methods and protocols defined for solving the problem of N players to compute jointly on an agreed function securely on the inputs without revealing them. In particular, there are N participants $p_1, p_2, \ldots, p_N$, and each one has private data $d_1, d_2, \ldots, d_N$. Participants want to compute the value of a public function on that private data: $F(d_1, d_2, \ldots, d_N)$ while keeping their own inputs secret.
For resolving this kind of problems, the Secure Multiparty Computation provides a protocol where no individual can see the other parties' data while distributing the data across multi parties. It enables the data scientists and analysts to compute privately on the distributed data without exposing it.



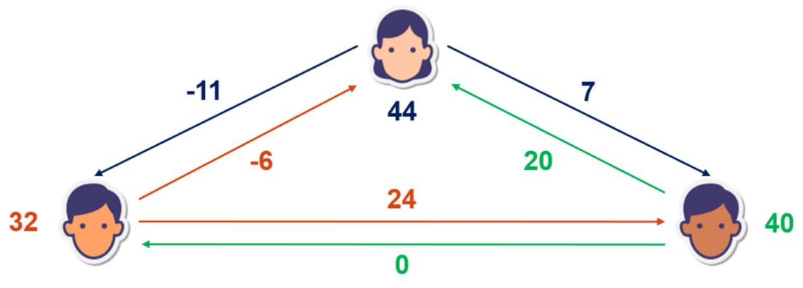
EXAMPLE
Imagine there are Alice, Bob and Charlie who are three co-workers that want to compute the average salary without revealing their individual salary to others. To perform such a computation, secure multiparty computation is implemented to calculate the average salary. The parties in a distributed manner jointly perform a function to calculate it without revealing the salary.



Using **additive secret sharing**, each salary is divided into randomly generated three pieces.

Alice	**40k $**	44	-11	7
Bob	**50k $**	-6	32	24
Charlie	**60k $**	20	0	40

Everyone keeps one of these secret pieces for himself and distributes the other two to others.



Alice	44	-6	20	**58k $**
Bob	-11	32	0	**21k $**
Charlie	7	24	40	**71k $**

During these operations, data are kept in encrypted form, broken up and distributed across parties in order to guarantee their security and privacy.

Total salary = 58k + 21k + 71k = 150k $
Average salary = 150k / 3 = 50k $

From the above data shared is possible to calculate the average salary. At the end of this computation, all that the three parties can learn is what they can learn from the output and their own input:

- Alice learns that her salary is less than the average one, which is 50k $.
- Bob learns that his salary is equal to the average one, which is 50k $.
- Charlie learns that his salary is more than the average one, which is 50k $.
- Every co-worker does not know the salary of others.

3. Advantages and disadvantages of SMPC

Here are some benefits of Secure Multiparty Computation:

1) **No trusted third party:** In Secure Multiparty Computation, we can share data in a distributed manner with different organizations without any third party and even the privacy of data will be preserved while sharing data.
2) **Data Privacy:** The confidential data of organizations can be shared for computation purposes. The concern of data privacy is provided by using secure multiparty computation, which keeps the data in use in encrypted form. Thus, the data is not revealed or compromised.

3) **High accuracy:** Secure Multiparty Computation provides highly accurate results for different computations using cryptography.
4) **Quantum safe:** The data shared between parties are safe against quantum attacks, as the data is broken up and encrypted when distributed among parties for computation.

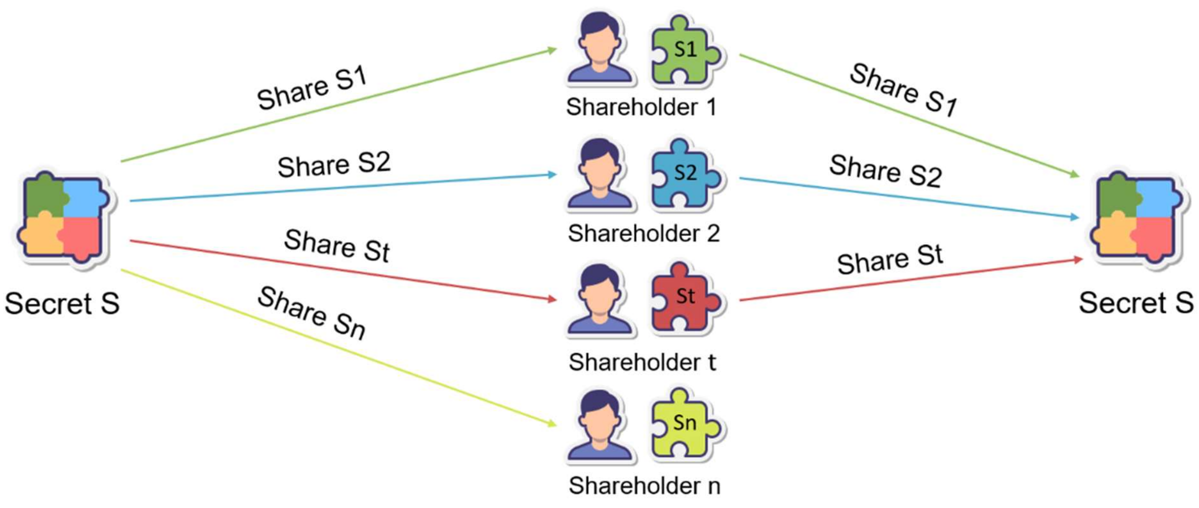
The main limitations of Secure Multiparty Computation are:

1) **Computational overhead:** To provide the security we need to generate the random numbers. The random number generation requires more computation overhead which slows down runtime.
2) **High communication costs:** Distributing the data to multiple parties for computation over the networks leads to higher costs of communication.

4. Shamir Secret Sharing

There are a number of techniques developed for Secure Multiparty Computation protocol construction having distinctive features. One of these is Shamir Secret Sharing.
Shamir Secret Sharing (SSS) is one of the most popular implementations of a secret sharing scheme created by Adi Shamir, a famous Israeli cryptographer, who also contributed to the invention of RSA algorithm.
It works as follows:

1) S is the secret that we wish to encode.
2) Divide S into n fragments called shares: S_1, S_2, …, S_n.
3) Distribute the secret S among multiple participants (shareholders), sending each of them a share.
4) The owner of S defines a threshold t. It is chosen in such a way that if we know less than t shares, then we will not be able to reconstruct the secret S. Only if we know t or more share, we can reconstruct S easily.



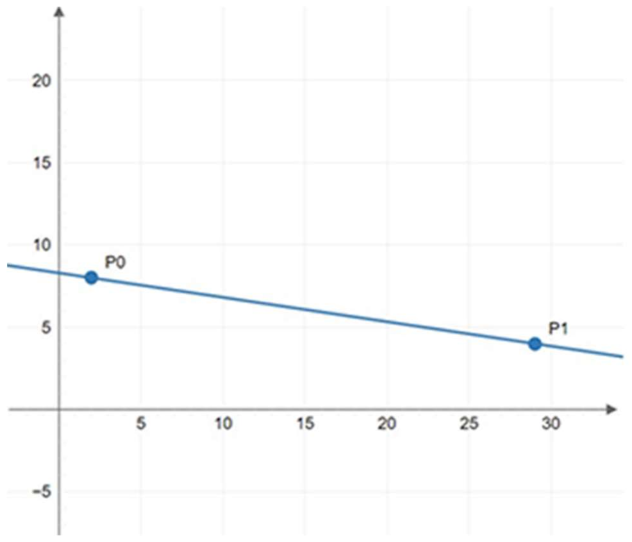
This is conventionally called **(t, n) threshold scheme**.

5. Polynomial interpolation

Shamir Secret Sharing is based on the mathematical concept of **polynomial interpolation** which states that a polynomial of degree t-1 can be reconstructed from the knowledge of t or more points, known to be lying on the curve.
To better understand this concept, let us start with a general polynomial of degree p can be expressed as follows:

$$P(x) = a_p x^p + a_{p-1} x^{p-1} + a_{p-2} x^{p-2} + \dots + a_2 x^2 + a_1 x + a_0$$

In the expression of P(x), the value a_0, a_1, a_2, …, a_p represent the coefficients of the polynomial. The construction of a polynomial requires the selection of these coefficients. Note that no values are actually substituted in for x because every term in the polynomial acts as a “placeholder” to store the coefficient values. Once the polynomial is generated, we can represent the curve of degree p by just p+1 points that lie on the curve. This follows from the polynomial interpolation principle.
In the image below, there is a straight line i.e. a curve of degree 1 that can be reconstructed if we have access to at least 2 unique points lying on it.



Straight line $y = a_1x + a_0$

In particular, to **generate shares** in a (t, n) secret sharing scheme, we can construct a random polynomial P(x) of degree t-1 by choosing random coefficient. Set the constant term in the polynomial (coefficient of zero-degree term) to be equal to the secret value S. To generate the n shares, randomly pick n points lying on the polynomial P(x). At the end, distribute the n shares among the participants of secret sharing scheme.
For **reconstructing the secret**, a minimum of t participants is required to pool their shares. After collecting t or more shares, use an interpolation algorithm such as Lagrange’s Interpolation to reconstruct the polynomial P'(x), from the shares. Then determine the value of the reconstructed polynomial for x = 0, so calculate P'(0). This value reveals the constant term of the polynomial which happens to be the original secret. Thus, the secret is reconstructed.

6. Advantages and disadvantages of SSS

Shamir Secret Sharing has useful properties, but also weaknesses that mean there are some situations where it should not be used.
Some useful properties include:

- **Secure:** The scheme has Information theoretic security.
- **Minimal:** The size of each piece does not exceed the size of the original data.
- **Extensible:** For any given threshold, shares can be dynamically added or deleted without affecting existing shares
- **Dynamic:** Security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares for the participants.
- **Flexible:** In organizations where hierarchy is important, each participant can be issued different numbers of shares according to their importance inside the organization. For instance, with a threshold of 3, the president could unlock the safe alone if given three shares, while three secretaries with one share each must combine their shares to unlock it.

Weaknesses include:

- **No verifiable secret sharing:** During the share reassembly process, SSS does not have a way to verify the correctness of each share being used. Verifiable secret sharing aims to verify that shareholders are honest and not submitting fake shares.
- **Single point of failure:** The secret must exist in one single place when It is split into shares, and again in one place when It's reassembled. These are attack points, and other schemes including multi signature eliminate at least one of these single points of failure.

7. Security problems

Unlike traditional cryptographic applications, such as encryption or signature, we must assume that the adversary in an MPC protocol is one of the players engaged in the system (or controlling internal parties). That corrupted party or parties may collude in order to breach the security of the protocol.
Let n be the number of parties in the protocol and t the number of parties who can be adversarial.
In general, the protocols are considered secure and reliable only when an honest majority is assumed: $t<n/2$.
Adversaries faced by the different protocols can be categorized according to how willing they are to deviate from the protocol. There are two types of adversaries, each giving rise to different forms of security:

- **Semi-Honest Security (passive):** In this case, it is assumed that corrupted parties merely cooperate to gather information out of the protocol, but don't deviate from the protocol specification. This is a naive adversary model, yielding weak security in real situations. However, protocols achieving this level of security prevent inadvertent leakage of information between (otherwise collaborating) parties and are thus useful if this is the only concern. In addition, protocols in the semi-honest model are quite efficient and are often an important first step for achieving higher levels of security.
- **Malicious Security (active):** In this case, the adversary may arbitrarily deviate from the protocol execution in its attempt to cheat. Protocols that achieve security in this model

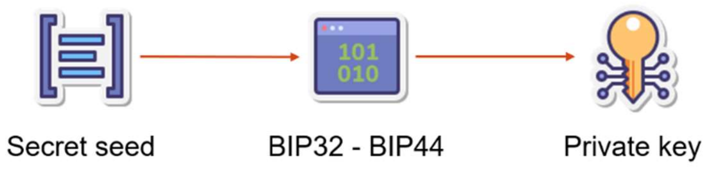
provide an extremely high security guarantee. The only thing that an adversary can do in the case of dishonest majority is to cause the honest parties to "abort" having detected cheating. If the honest parties do obtain output, then they are guaranteed that It is correct. Their privacy is always preserved.

Security against active adversaries typically leads to a reduction in efficiency that leads to covert security, a relaxed form of active security. Covert security captures more realistic situations, where active adversaries are willing to cheat but only if they are not caught. For example, their reputation could be damaged, preventing future collaboration with other honest parties. Thus, protocols that are covertly secure provide mechanisms to ensure that, if some of the parties do not follow the instructions, then it will be noticed with high probability, say 75% or 90%. In a way, covert adversaries are active ones forced to act passively due to external non-cryptographic concerns. This mechanism sets a bridge between both models in the hope of finding protocols which are efficient and secure enough in practice.

8. Wallet management

In the Bitcoin blockchain, transactions are authenticated by those who create them using digital signatures and are subsequently validated by the miners. The signer uses his secret private key to sign. This key must be kept in a safe place to prevent cybercriminals from getting it and starting to create fraudulent transactions.

For Bitcoin and other similar cryptocurrencies, private keys are generated deterministically from a secret seed. Algorithms that perform such generation are BIP32 and BIP44. The algorithm will always generate the same private key from the same seed.



Private key is then used by various users to access their deterministic Bitcoin wallet. Therefore, it is clear that the problem of protecting the private key that validates a transaction is reduced to the problem of protecting the secret seed that generated it. Naturally, the loss of the private key or of secret seed would compromise access to the deterministic wallet and all the Bitcoins contained within it.

To protect the secret seed and its deterministic wallet, we use Shamir Secret Sharing. In particular, the secret seed should be fragmented using a (t, n) threshold scheme where the seed is divided into n shares which are saved on n different devices and t or more shares will be needed to obtain the secret seed. Once the secret seed has been reconstructed from a sufficient number of shares, it is possible to reuse the deterministic algorithm to reconstruct the private key for accessing the Bitcoin wallet. Obviously, the devices used to store shares must be protected and kept in a safe place.